CLAIMS

1. A method for dynamic recompilation of a program, comprising:

identifying binary code for a program;

obtaining a portion of the binary code;

executing the portion of the binary code while optimizing the portion of the binary code, the executing identifying dynamic changes in flow to enable additional portions of the binary code to be obtained and executed; and

saving the executed and optimized portion of the binary code and any additional portions of the binary code to an optimized binary code file for the program.

2. The method of claim 1, further comprising:

continuing obtaining and executing portions of the binary code until all portions of the binary code have been saved to the optimized binary code file for the program.

3. The method of claim 2, further comprising:

executing the optimized binary code file for the program;

detecting a missing additional portion associated with a dynamic change in flow detected during execution of a portion of the optimized binary code file for the program;

obtaining the missing additional portion from the binary code for the program;

executing the missing additional portion; and

saving the executed missing additional portion to the optimized binary code file for the program.

4. The method of claim 1, wherein the dynamic changes in flow include a jump instruction.

5. The method of claim 1, wherein the optimizing is configured to optimize the portion of the binary code for a new hardware architecture.

6. A method for dynamic recompilation of a program, comprising:

a) identifying binary code for a program;

b) obtaining a portion of the binary code;

c) executing the portion of the binary code while optimizing the portion of the binary code, the executing identifying dynamic changes in flow to enable additional portions of the binary code to be obtained and executed;

d) saving the executed and optimized portion of the binary code and any additional portions of the binary code to an optimized binary code file for the program; and

e) repeating operations b), c), and d) until all portions of the binary code have been saved to the optimized binary code file for the program.

7. The method of claim 6, further comprising:

executing the optimized binary code file for the program;

detecting a missing additional portion associated with a dynamic change in flow detected during execution of a portion of the optimized binary code file for the program;

obtaining the missing additional portion from the binary code for the program;

executing the missing additional portion; and

saving the executed missing additional portion to the optimized binary code file for the program.

8. The method of claim 7, wherein the dynamic changes in flow include a jump instruction.

9. The method of claim 6, wherein the optimizing is configured to optimize the portion of the binary code for a new hardware architecture.

10. Computer readable media containing program instructions for dynamic recompilation of a program, the computer readable media comprising:

program instructions for identifying binary code for a program;

program instructions for obtaining a portion of the binary code;

program instructions for executing the portion of the binary code while optimizing the portion of the binary code, the executing identifying dynamic changes in flow to enable additional portions of the binary code to be obtained and executed; and

program instructions for saving the executed and optimized portion of the binary code and any additional portions of the binary code to an optimized binary code file for the program.

11. The computer readable media of claim 10, further comprising:

program instructions for continuing obtaining and executing portions of the binary code until all portions of the binary code have been saved to the optimized binary code file for the program.

12. The computer readable media of claim 11, further comprising:

program instructions for executing the optimized binary code file for the program;

program instructions for detecting a missing additional portion associated with a dynamic change in flow detected during execution of a portion of the optimized binary code file for the program;

program instructions for obtaining the missing additional portion from the binary code for the program;

program instructions for executing the missing additional portion; and

program instructions for saving the executed missing additional portion to the optimized binary code file for the program.

13. The computer readable media of claim 10, wherein the dynamic changes in flow include a jump instruction.

14. The method of claim 10, wherein the optimizing is configured to optimize the portion of the binary code for a new hardware architecture.

15. Computer readable media containing program instructions for dynamic recompilation of a program, the computer readable media comprising:

a) program instructions for identifying binary code for a program;

b) program instructions for obtaining a portion of the binary code;

c) program instructions for executing the portion of the binary code while optimizing the portion of the binary code, the executing identifying dynamic changes in flow to enable additional portions of the binary code to be obtained and executed;

d) program instructions for saving the executed and optimized portion of the binary code and any additional portions of the binary code to an optimized binary code file for the program; and

e) program instructions for repeating program instructions b), c), and d) until all portions of the binary code have been saved to the optimized binary code file for the program.

16. The computer readable media of claim 15, further comprising:

program instructions for executing the optimized binary code file for the program;

program instructions for detecting a missing additional portion associated with a dynamic change in flow detected during execution of a portion of the optimized binary code file for the program;

program instructions for obtaining the missing additional portion from the binary code for the program;

program instructions for executing the missing additional portion; and

program instructions for saving the executed missing additional portion to the optimized binary code file for the program.

17. The computer readable media of claim 15, wherein the dynamic changes in flow include a jump instruction.

18. The method of claim 15, wherein the optimizing is configured to optimize the portion of the binary code for a new hardware architecture.